L Number	Hits	Search Text	DB	Time stamp
1	166	(717/109).CCLS.	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:21
2	85	(717/110).CCLS.	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:21
3	218	(717/116).CCLS.	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:21
4	251	(717/124).CCLS.	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:21
5	143	(717/125).CCLS.	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:21
6	114	(717/126).CCLS.	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:21
7	239	(717/127).CCLS.	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:21
8	208	(717/140).CCLS.	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:22
9	57	(717/152).CCLS.	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:22
10	2	((717/109).CCLS.) and execut$4 and native adj code and compil$4 and edit$4	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:23
11	1	((717/110).CCLS.) and execut$4 and native adj code and compil$4 and edit$4	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:23
12	3	((717/116).CCLS.) and execut$4 and native adj code and compil$4 and edit$4	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:24
13	2	((717/124).CCLS.) and execut$4 and native adj code and compil$4 and edit$4	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:24
14	0	((717/125).CCLS.) and execut$4 and native adj code and compil$4 and edit$4	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:24
15	1	((717/126).CCLS.) and execut$4 and native adj code and compil$4 and edit$4	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:24
16	5	((717/127).CCLS.) and execut$4 and native adj code and compil$4 and edit$4	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:25
17	2	((717/140).CCLS.) and execut$4 and native adj code and compil$4 and edit$4	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:26

18	131	execut$4 and native adj code and compil$4 and edit$4	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:26
19	5	(execut$4 and native adj code and compil$4 and edit$4) and object and class and intermediate adj language	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/09/02 12:26

	U	1	Document ID	Issue Date	Pages	Title	Current OR
1	☒	☐	US 6760903 B1	20040706	121	Coordinated application monitoring in a distributed computing environment	717/130
2	☐	☐	US 6738967 B1	20040518	22	Compiling for multiple virtual machines targeting different processor architectures	717/146
3	☒	☐	US 6721941 B1	20040413	122	Collection of timing and coverage data through a debugging interface	717/127
4	☒	☐	US 6668325 B1	20031223	62	Obfuscation techniques for enhancing software security	713/194
5	☒	☐	US 4667290 A	19870519	33	Compilers using a universal intermediate language	717/147

	Current XRef	Retrieval Classif	Inventor	S	C	P	2	3	4	5
1			Morshed, Farokh et al.	☐	☐	☐	☐	☐	☐	☐
2			Radigan, James J.	☒	☐	☐	☐	☐	☐	☐
3	709/217; 714/38; 717/124; 717/126; 717/128; 717/129; 717/130; 717/131; 719/328		Morshed, Farokh et al.	☐	☐	☐	☐	☐	☐	☐
4	713/200		Collberg, Christian Sven et al.	☐	☐	☐	☐	☐	☐	☐
5	713/1; 717/114; 717/143		Goss, Clinton et al.	☐	☐	☐	☐	☐	☐	☐

	U	1	Document ID	Issue Date	Pages	Title	Current OR
1	☐	☐	US 6721941 B1	20040413	122	Collection of timing and coverage data through a debugging interface	717/127
2	☐	☐	US 6643842 B2	20031104	45	Byte code instrumentation	717/130
3	☐	☐	US 6634019 B1	20031014	71	Toggling software characteristics in a fault tolerant and combinatorial software environment system, method and medium	717/127
4	☐	☐	US 6567974 B1	20030520	22	Small memory footprint system and method for separating applications within a single virtual machine	717/151
5	☐	☐	US 6557168 B1	20030429	22	System and method for minimizing inter-application interference among static synchronized methods	717/151

	Current XRef	Retrieval Classif	Inventor	S	C	P	2	3	4	5
1	709/217; 714/38; 717/124; 717/126; 717/128; 717/129; 717/130; 717/131; 719/328		Morshed, Farokh et al.	☒	☐	☐	☐	☐	☐	☐
2	714/35; 717/118; 717/127		Angel, David J. et al.	☒	☐	☐	☐	☐	☐	☐
3	706/13		Rice, Todd M. et al.	☒	☐	☐	☐	☐	☐	☐
4	717/118; 717/127; 717/161		Czajkowski, Grzegorz J.	☒	☐	☐	☐	☐	☐	☐
5	709/400; 717/127		Czajkowski, Grzegorz J.	☒	☐	☐	☐	☐	☐	☐

P◉RTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login
Search: ◉ The ACM Digital Library ○ The Guide
execute native code and compile SEARCH

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **execute native code** and **compile** Found **38,632** of **141,680**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Performance measurement of dynamically compiled Java executions
Tia Newhall, Barton P. Miller
June 1999 **Proceedings of the ACM 1999 conference on Java Grande**
Full text available: pdf(1.54 MB) Additional Information: full citation, references, citings, index terms

Keywords: Java, dynamic compilation, performance profiling tool

2 Native code compilation of Erlang's bit syntax
Per Gustafsson, Konstantinos Sagonas
October 2002 **Proceedings of the 2002 ACM SIGPLAN workshop on Erlang**
Full text available: pdf(196.81 KB) Additional Information: full citation, abstract, references, citings

Erlang's bit syntax caters for flexible pattern matching on bit streams (objects known as *binaries*). Binaries are nowadays heavily used in typical Erlang applications such as protocol programming, which in turn has created a need for efficient support of the basic operations on binaries.To this effect, we describe a scheme for efficient native code compilation of Erlang's bit syntax. The scheme relies on *partial translation* for avoiding code explosion, an ...

3 Proxy compilation of dynamically loaded Java classes with MoJo
Matt Newsome, Des Watson
June 2002 **ACM SIGPLAN Notices , Proceedings of the joint conference on Languages, compilers and tools for embedded systems: software and compilers for embedded systems**, Volume 37 Issue 7
Full text available: pdf(358.62 KB) Additional Information: full citation, abstract, references, index terms

Interest in Java implementations for resource-constrained environments such as embedded systems has been tempered by concerns regarding its efficiency. Current native compilers for Java offer dramatic increases in efficiency, but have poor support for dynamically-loaded classes, which are typically served by slow interpreters or JIT compilers, the code-size of this latter utterly mismatching the resource constraints of the system.After a brief survey of Ahead-of-Time compilers for Java, we prese ...

Keywords: AOT, JIT, adaptive compilation, ahead-of-time, dynamic class loading, hotspot, just-in-time, native compilation, proxy compilation, remote compilation

4 Java bytecode to native code translation: the caffeine prototype and preliminary results
Cheng-Hsueh A. Hsieh, John C. Gyllenhaal, Wen-mei W. Hwu
December 1996 **Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture**
Full text available: pdf(1.03 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

The Java bytecode language is emerging as a software distribution standard. With major vendors committed to porting the Java run-time environment to their platforms, programs in Java bytecode are expected to run without modification on multiple platforms. These first generation run-time environments rely on an interpreter to bridge the gap between the bytecode instructions and the native hardware. This interpreter approach is sufficient for specialized applications such as Internet browsers wher ...

5 Adaptive code unloading for resource-constrained JVMs
Lingli Zhang, Chandra Krintz
June 2004 **ACM SIGPLAN Notices , Proceedings of the 2004 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools**, Volume 39 Issue 7
Full text available: pdf(204.29 KB) Additional Information: full citation, abstract, references, index terms

Compile-only JVMs for resource-constrained embedded systems have the potential for using device resources more efficiently than interpreter-only systems since compilers can produce significantly higher quality code and code can be stored and reused for future invocations. However, this additional storage requirement for reuse of native code bodies, introduces memory overhead not imposed in interpreter-based systems.In this paper, we present a Java Virtual Machine (JVM) extension for adaptive cod ...

Keywords: JIT, JVM, code unloading, code-size reduction, resource-constrained devices

6 Techniques for obtaining high performance in Java programs
Iffat H. Kazi, Howard H. Chen, Berdenia Stanley, David J. Lilja
September 2000 **ACM Computing Surveys (CSUR)**, Volume 32 Issue 3
Full text available: pdf(816.13 KB) Additional Information: full citation, abstract, references, citings, index terms

This survey describes research directions in techniques to improve the performance of programs written in the Java programming language. The standard technique for Java execution is interpretation, which provides for extensive portability of programs. A Java interpreter dynamically executes Java bytecodes, which comprise the instruction set of the Java Virtual Machine (JVM). Execution time performance of Java programs can be improved through compilation, possibly at the expense of portabili ...

Keywords: Java, Java virtual machine, bytecode-to-source translators, direct compilers, dynamic compilation, interpreters, just-in-time compilers

7 A high performance Erlang system
Erik Johansson, Mikael Pettersson, Konstantinos Sagonas
September 2000 **Proceedings of the 2nd ACM SIGPLAN international conference on Principles and practice of declarative programming**
Full text available: pdf(320.62 KB) Additional Information: full citation, references, citings, index terms

8 Multitasking without comprimise: a virtual machine evolution
Grzegorz Czajkowski, Laurent Daynés
October 2001 **ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications**, Volume 36 Issue 11
Full text available: pdf(220.97 KB) Additional Information: full citation, abstract, references, citings, index terms

The multitasking virtual machine (called from now on simply MVM) is a modification of the Java virtual machine. It enables safe, secure, and scalable multitasking. Safety is achieved by strict isolation of application from one another. Resource control augment security by preventing some denial-of-service attacks. Improved scalability results from an aggressive application of the main design principle of MVM: share as much of the runtime as possible among applications and replicate everything el ...

Keywords: Java virtual machine, application isolation, native code execution, resource control

9 A brief history of just-in-time
John Aycock
June 2003 **ACM Computing Surveys (CSUR)**, Volume 35 Issue 2
Full text available: pdf(171.09 KB) Additional Information: full citation, abstract, references, index terms

Software systems have been using "just-in-time" compilation (JIT) techniques since the 1960s. Broadly, JIT compilation includes any translation performed dynamically, after a program has started execution. We examine the motivation behind JIT compilation and constraints imposed on JIT compilation systems, and present a classification scheme for such systems. This classification emerges as we survey forty years of JIT work, from 1960--2000.

Keywords: Just-in-time compilation, dynamic compilation

10 A framework for efficient reuse of binary code in Java
Pramod G. Joisha, Samuel P. Midkiff, Mauricio J. Serrano, Manish Gupta
June 2001 **Proceedings of the 15th international conference on Supercomputing**
Full text available: pdf(419.49 KB) Additional Information: full citation, abstract, references, index terms

This paper presents a compilation framework that enables efficient sharing of executable code across distinct Java Virtual Machine (JVM) instances. High-performance JVMs rely on run-time compilation, since static compilation cannot handle many dynamic features of Java. These JVMs suffer from large memory footprints and high startup costs, which are serious problems for embedded devices (such as hand held personal digital assistants and cellular phones) and scalable servers. A recently propose ...

11 LLVA: A Low-level Virtual Instruction Set Architecture
Vikram Adve, Chris Lattner, Michael Brukman, Anand Shukla, Brian Gaeke
December 2003 **Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture**
Full text available: pdf(196.08 KB) Publisher Site Additional Information: full citation, abstract, index terms

A virtual instruction set architecture (V-ISA) implementedvia a processor-specific software translation layercan provide great flexibility to processor designers. Recentexamples such as Crusoe and DAISY, however, haveused existing hardware instruction sets as virtual ISAs,which complicates translation and optimization. In fact,there has been little research on specific designs for a virtualISA for processors. This paper proposes a novel virtualISA (LLVA) and a translation strategy for implementi ...

12 Optimizing ML with run-time code generation
Peter Lee, Mark Leone
May 1996 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1996 conference on Programming language design and implementation**, Volume 31 Issue 5
Full text available: pdf(1.34 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe the design and implementation of a compiler that automatically translates ordinary programs written in a subset of ML into code that generates native code at run time. Run-time code generation can make use of values and invariants that cannot be exploited at compile time, yielding code that is often superior to statically optimal code. But the cost of optimizing and generating code at run time can be prohibitive. We demonstrate how compile-time specialization can reduce the cost of r ...

13 Practicing JUDO: Java under dynamic optimizations
Michał Cierniak, Guei-Yuan Lueh, James M. Stichnoth
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation**, Volume 35 Issue 5
Full text available: pdf(190.06 KB) Additional Information: full citation, abstract, references, citings, index terms

A high-performance implementation of a Java Virtual Machine (JVM) consists of efficient implementation of Just-In-Time (JIT) compilation, exception handling, synchronization mechanism, and garbage collection (GC). These components are tightly coupled to achieve high performance. In this paper, we present some static anddynamic techniques implemented in the JIT compilation and exception handling of the Microprocessor Research Lab Virtual Machine (MRL VM), ...

14 Compilation and run-time systems: DELI: a new run-time control point
Giuseppe Desoli, Nikolay Mateev, Evelyn Duesterwald, Paolo Faraboschi, Joseph A. Fisher
November 2002 **Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture**
Full text available: pdf(1.27 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

The Dynamic Execution Layer Interface (DELI) offers the following unique capability: it provides fine-grain control over the execution of programs, by allowing its clients to observe and optionally manipulate every single instruction---at run time---just before it runs. DELI accomplishes this by opening up an interface to the layer between the execution of software and hardware. To avoid the slowdown, DELI caches a private copy of the executed code and always runs out of its own private cache.In ...

15 Improving Java performance using hardware translation
Ramesh Radhakrishnan, Ravi Bhargava, Lizy K. John
June 2001 **Proceedings of the 15th international conference on Supercomputing**
Full text available: pdf(254.91 KB) Additional Information: full citation, abstract, references, citings, index terms

State of the art Java Virtual Machines with Just-In-Time (JIT) compilers make use of advanced compiler techniques, run-time profiling and adaptive compilation to improve performance. However, these techniques for alleviating performance bottlenecks are more effective in long running workloads, such as server applications. Short running Java programs, or client workloads, spend a large fraction of their execution time in compilation instead of useful execution when run using JIT compilers. In ...

16 Joeq: a virtual machine and compiler infrastructure
John Whaley
June 2003 **Proceedings of the 2003 workshop on Interpreters, Virtual Machines and Emulators**
Full text available: pdf(206.13 KB) Additional Information: full citation, abstract, references, citings, index terms

Joeq is a virtual machine and compiler infrastructure designed to facilitate research in virtual machine technologies such as Just-In-Time and Ahead-Of-Time compilation, advanced garbage collection techniques, distributed computation, sophisticated scheduling algorithms, and advanced run time techniques. Joeq is entirely implemented in Java, leading to reliability, portability, maintainability, and efficiency. It is also language-independent, so code from any supported language can be seamlessly ...

17 Dynamic Adaptive compilation: Design, implementation and evaluation of adaptive recompilation with on-stack replacement
Stephen J. Fink, Feng Qian
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**
Full text available: pdf(1.02 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

Modern virtual machines often maintain multiple compiled versions of a method. An on-stack replacement (OSR) mechanism enables a virtual machine to transfer execution between compiled versions, even while a method runs. Relying on this mechanism, the system can exploit powerful techniques to reduce compile time and code space, dynamically de-optimize code, and invalidate speculative optimizations.This paper presents a new, simple, mostly compiler-independent mechanism to transfer execution into ...

18 Object allocation and dynamic compilation in MultithreadSmalltalk
Kazuhiro Ogata, Norihisa Doi
April 1994 **Proceedings of the 1994 ACM symposium on Applied computing**
Full text available: pdf(476.46 KB) Additional Information: full citation, references, index terms

Keywords: MultithreadSmalltalk, Smalltalk, dynamic compilation, multiprocessor, object allocation

19 Implementing jalapeño in Java
Bowen Alpern, C. R. Attanasio, Anthony Cocchi, Derek Lieber, Stephen Smith, Ton Ngo, John J. Barton, Susan Flynn Hummel, Janice C. Sheperd, Mark Mergen
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Full text available: pdf(1.57 MB) Additional Information: full citation, abstract, references, citings, index terms

Jalapeño is a virtual machine for Java™ servers written in Java.A running Java program involves four layers of functionality: the user code, the virtual-machine, the operating system, and the hardware. By drawing the Java / non-Java boundary below the virtual machine rather than above it, Jalapeño reduces the boundary-crossing overhead and opens up more opportunities for optimization.To get Jalapeño started, a boot image of a ...

20 Profile-guided optimization across process boundaries
Erik Johansson, Sven-Olof Nyström
January 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN workshop on Dynamic and adaptive compilation and optimization**, Volume 35 Issue 7
Full text available: pdf(911.89 KB) Additional Information: full citation, abstract, references, citings, index terms

We describe a profile-driven compiler optimization technique for *inter-process optimization*, which dynamically inlines the effects of sending messages. Profiling is used to find optimization opportunities, and to dynamically trigger recompilation and optimization at run-time. We apply the optimization technique on the concurrent programming language ERLANG, letting recompilation take place in a separate ERLANG process, and taking advantage of the facilities provided by ERLANG to dynami ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next